Memory Lower Bounds for Randomized Collaborative Search and Applications to Biology

Ofer Feinerman, Amos Korman

▶ **To cite this version:**

Ofer Feinerman, Amos Korman. Memory Lower Bounds for Randomized Collaborative Search and Applications to Biology. Distributed Computing - 26th International Symposium, DISC 2012, Oct 2012, Salvador, Brazil. 10.1007/978-3-642-33651-5_5 . hal-01241102

HAL Id: hal-01241102
https://inria.hal.science/hal-01241102

Submitted on 6 Jan 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Memory Lower Bounds for Randomized Collaborative Search and Applications to Biology

Ofer Feinerman* Amos Korman†

Abstract

Initial knowledge regarding group size can be crucial for collective performance. We study this relation in the context of the *Ants Nearby Treasure Search (ANTS)* problem [32], which models natural cooperative foraging behavior such as that performed by ants around their nest. In this problem, k (probabilistic) agents, initially placed at some central location, collectively search for a treasure on the two-dimensional grid. The treasure is placed at a target location by an adversary and the goal is to find it as fast as possible as a function of both k and D, where D is the (unknown) distance between the central location and the target. It is easy to see that $T = \Omega(D + D^2/k)$ time units are necessary for finding the treasure. Recently, it has been established that $O(T)$ time is sufficient if the agents know their total number k (or a constant approximation of it), and enough memory bits are available at their disposal [32]. In this paper, we establish lower bounds on the agent memory size required for achieving certain running time performances. To the best our knowledge, these bounds are the first non-trivial lower bounds for the memory size of probabilistic searchers. For example, for every given positive constant ϵ, terminating the search by time $O(\log^{1-\epsilon} k \cdot T)$ requires agents to use $\Omega(\log \log k)$ memory bits. Such distributed computing bounds may provide a novel, strong tool for the investigation of complex biological systems.

Keywords: search algorithms; memory bounds; advice complexity; mobile robots; speed-up; cow-path problem; online algorithms; quorum sensing; social insects; central place foraging, ants.

*The Louis and Ida Rich Career Development Chair, The Weizmann Institute of Science, Rehovot, Israel. E-mail: feinermanofer@gmail.com. Supported by the Israel Science Foundation (grant 1694/10).

†CNRS and University Paris Diderot, Paris, France. E-mail: amos.korman@liafa.jussieu.fr. Supported in part by the ANR projects DISPLEXITY and PROSE, and by the INRIA project GANG.

1 Introduction

Background and Motivation: Individuals in biological groups assemble in groups that allow them, among other things, to monitor and react to relatively large environments. For this, individuals typically disperse over length scales that are much larger than those required for communication. Thus, collecting knowledge regarding larger areas dictates a dispersion that may come at the price of group coordination and efficient information sharing. A possible solution involves the use of designated, localized areas where individuals convene to share information and from which they then disperse to interact with the environment. Indeed, there are numerous examples for such convention areas in the biological world. Cells of the immune system undergo a collective activation, differentiation and maturation process away from the site of infection and within compact lymph nodes [24]. Birds are known to travel long distances from their feeding grounds to communal sleeping area where they were shown to share information regarding food availability [56, 57]. Here we focus, on a third example, that of collective *central place foraging* [39, 23] where a group of animals leave a central location (e.g., a nest) to which they then retrieve collected food items. Here as well, the localized nest area enables efficient communication. Ants, for example, were shown to share information within their nest regarding food availability and quality outside it [10, 58]. This information is then used as a means of regulating the foraging efforts.

One piece of information that may be available to a localized group is its *size*. Group size may be used to reach collective decisions which then affect the subsequent behavioral repertoire of the individuals. The most prevalent example is that of *quorum sensing*; a binary estimate of group size or density. Such threshold measurements are exhibited by a multitude of biological systems such as bacteria [49], amoeba [53], T-cells of the immune system [9, 31] and social insects [42]. The quorum sensing process constitutes a first decision step that may lead to cell differentiation and divergent courses of action. Going beyond quorum sensing: there are evidences for higher resolution estimates of group size in, for example, wild dogs where multiple hunting tactics are employed in correlation with increasing numbers of participating individuals [52].

Here, we focus on the potential benefits of estimating group size in the context of collective central place foraging. Ants, for example, engage in this behavior in a cooperative manner - individuals search for food items around the nest and share any findings. Clearly, due to competition and other time constrains, food items must be found relatively fast. Furthermore, finding food not only fast but also in proximity to the central location holds numerous advantages at both the search and the retrieval stages. Such advantages include, for example, decreasing predation risk [25], and increasing the rate of food collection once a large quantity of food is found [39, 23]. Intuitively, the problem at hand is distributing searchers within bounded areas around the nest while minimizing overlaps. Ants may possibly exchange information inside the nest, however, once they are out, minimizing search overlaps decreases the rate of communication via pairwise interaction, in some species, to a seemingly negligible degree [23].

It was previously shown that the efficiency of collective central place foraging may be enhanced by initial knowledge regarding group size [32]. More specifically, that paper introduces the *Ants Nearby Treasure Search (ANTS)* problem, which models the aforementioned central place foraging setting. In this problem, k (probabilistic) agents, initially placed at some central location, collectively search for a treasure in the two-dimensional grid. The treasure is placed at a target location by an adversary and the goal is to find it as fast as possible as a function of both k and D, where D is the (unknown) distance between the central location and the target. Once the agents initiate the search they cannot communicate between themselves. Based on volume considerations, it is an easy observation that the expected running time of any algorithm is $\Omega(D + D^2/k)$. It was established in [32] that the knowledge of a constant approximation of k allows the agents to find the treasure in asymptotically optimal expected time, namely, $O(D + D^2/k)$. On the other hand, the lack of any information of k prevents them from reaching expected time that is higher than optimal by a factor of $O(\log k)$. That work also establishes lower bounds on the competitiveness of the algorithm in the particular

case where some given approximation to k is available to all nodes.

In this work, we simulate the initial step of information (e.g., regarding group size) sharing within the nest by using the abstract framework of *advice* (see, e.g., [11, 18, 21]). That is, we model the preliminary process for gaining knowledge about k (e.g., at the central location) by means of an *oracle* that assigns advice to agents. To measure the amount of information accessible to agents, we analyze the *advice size*, that is, the maximum number of bits used in an advice. Since we are mainly interested in lower bounds on the advice size required to achieve a given competitive ratio, we apply a liberal approach and assume a highly powerful oracle. More specifically, even though it is supposed to model a distributed (probabilistic) process, we assume that the oracle is a centralized probabilistic algorithm (almost unlimited in its computational power) that can assign each agent with a different advice. Note that, in particular, by considering identifiers as part of the advice, our model allows to relax the assumption that all agents are identical and to allow agents to be of several types. Indeed, in the context of ants, it has been established that ants on their first foraging bouts execute different protocols than those that are more experienced [54].

The main technical results of this paper deal with lower bounds on the advice size. For example, with the terminology of advice, [32] showed that advice of size $O(\log\log k)$ bits is sufficient to obtain an $O(1)$-competitive algorithm. We prove that this bound is tight. In fact, we show a much stronger result, that is, that advice of size $\Omega(\log\log k)$ is necessary even for achieving competitiveness which is as large as $O(\log^{1-\epsilon} k)$, for every given positive constant ϵ. On the other extremity, we show that $\Omega(\log\log\log k)$ bits of advice are necessary for being $O(\log k)$-competitive, and that this bound is tight. In addition, we exhibit lower bounds on the corresponding advice size for a range of intermediate competitivenesses.

Observe that the advice size bounds from below the number of memory bits used by an agent, as this amount of bits in required merely for storing some initial information. In general, from a purely theoretical point of view, analyzing the memory required for efficient search is a central theme in computer science [45, 48], and is typically considered to be difficult. To the best of our knowledge, this paper is the first to exhibit non-trivial lower bounds for the memory size of probabilistic agents in the context of search problems.

From a high level perspective, we hope to illustrate that distributed computing can potentially provide a novel and efficient methodology for the study of highly complex, cooperative biological ensembles. Indeed, if experiments that fit our setting reveal that the ants' search is time efficient, in the sense detailed above, then our theoretical results can provide some insight on the memory ants use for this task. A detailed discussion of this approach is given in Section 5.

Our results: The main technical results deal with lower bounds on the advice size. Our first result is perhaps the most surprising one. It says not only that $\Omega(\log\log k)$ bits of advice are required to obtain an $O(1)$-competitive algorithm, but that roughly this amount is necessary even for achieving competitiveness which is as large as $O(\log^{1-\epsilon} k)$, for every given positive constant ϵ. This result should be put in contrast to the fact that with no advice at all, one can obtain a search algorithm whose competitiveness is slightly more than logarithmic [32].

Theorem 1.1 *There is no search algorithm that is $O(\log^{1-\epsilon} k)$-competitive for some fixed positive ϵ, using advice of size $o(\log\log k)$.*

On the other extremity, we show that $\Omega(\log\log\log k)$ bits of advice are necessary for constructing an $O(\log k)$-competitive algorithm, and we prove that this bound on the advice is in fact tight.

Theorem 1.2 *There is no $O(\log k)$-competitive search algorithm , using advice of size $\log\log\log k - \omega(1)$. On the other hand, there exists an $O(\log k)$-competitive search algorithm using advice of size $\log\log\log k + O(1)$.*

	Competitiveness	Advice size
Tight bound	$O(1)$	$\Theta(\log\log k)$
Tight bound	$O(\log^{1-\epsilon} k) \quad 0<\epsilon<1$	$\Theta(\log\log k)$
Lower bound	$\log k/2^{\log^{\epsilon}\log k} \quad 0<\epsilon<1$	$\log^{\epsilon}\log k - O(1)$
Tight bound	$O(\log k)$	$\log\log\log k + \Theta(1)$
Upper bound [32]	$O(\log^{1+\epsilon} k)$	zero

Table 1: Bounds on the advice for given competitiveness

Finally, we also exhibit lower bounds for the corresponding advice size for a range of intermediate competitivenesses.

Theorem 1.3 *Consider a $\Phi(k)$-competitive search algorithm using advice of size $\Psi(k)$. Then, $\Phi(k) = \Omega(\log k/2^{\Psi(k)})$, or in other words, $\Psi(k) = \log\log k - \log\Phi(k) - O(1)$. In particular, if $\Phi(k) = \frac{\log k}{2^{\log^{\epsilon}\log k}}$, then $\Psi(k) = \log^{\epsilon}\log k - O(1)$.*

Our results on the advice complexity are summarized in Table 1. As mentioned, our lower bounds on the advice size are also lower bounds on the memory size of agents. To the best of our knowledge, this paper is the first to exhibit non-trivial lower bounds for the memory size of probabilistic agents in the context of search problems.

Related Work: Our current work falls within the framework of natural algorithms, a recent attempt to study biological phenomena from an algorithmic perspective [1, 7, 14, 32].

The notion of advice is central in computer science (in fact, checking membership in NP-languages can be viewed as computing with advice). In particular, the concept of advice and its impact on various computations has recently found various applications in distributed computing. In this context, the main measure used is the advice size. It is for instance analyzed in frameworks such as proof labeling [34, 35], broadcast [18], local computation of MST [21], graph coloring [19] and graph searching by a single robot [11]. Very recently, it has also been investigated in the context of online algorithms [8, 17].

Collective search is a classical problem that has been extensively studied in different contexts (for a more comprehensive summary refer to [32]). Social foraging theory [22] and central place foraging typically deal with optimal resource exploitation strategies between competing or cooperating individuals. Actual collective search trajectories of non-communicating agents have been studied in the physics literature (e.g., [43, 59]). Reynolds [43] achieves optimal speed up through overlap reduction which is obtained by sending searchers on near -straight disjoint lines to infinity. This must come at the expense of finding proximal treasures. Harkness and Maroudas [23] combined field experiments with computer simulations of a semi-random collective search and suggest substantial speed ups as group size increases. The collective search problem has further been studied from an engineering perspective (e.g., [41]). In this case, the communication between agents (robots) or their computational abilities are typically unrestricted. These works put no emphasis on finding nearby treasures fast. Further, there is typically no reference to group size or its knowledge by the agents.

In the theory of computer science, the exploration of graphs using mobile agents is a central question. Most of the research for graph exploration is concerned with the case of a single deterministic agent exploring a finite graph, see, e.g., [2, 6, 28, 29, 30, 40, 45]. The more complex setting on using multiple identical agents has received much less attention. Exploration by deterministic multiple agents was studied in, e.g., [4, 27, 33]. In general, one of the main challenges in search problems is the establishment of memory bounds. For example, the question of whether a single agent can explore all finite undirected graphs using logarithmic

memory was open for a long time; answering it to the affirmative [45] established an equality between the classes of languages SL and L. As another example, it was proved in [48] that no finite set of constant memory agents can explore all graphs. To the best of our knowledge, the current paper is the first paper establishing non-trivial lower bounds for the memory of randomized searching agents with respect to given time constrains.

The simplest (and most studied) probabilistic search algorithm is the random walk. In particular, several studies analyzing the speed-up measure for k-random walkers have recently been published. In these papers, a speed-up of $\Omega(k)$ is established for various finite graph families, including, e.g., expenders and random graphs [3, 16, 12]. In contrast, for the two-dimensional n-node grid, as long as k is polynomial in n, the speed up is only logarithmic in k. The situation with infinite grids is even worse. Specifically, though the k-random walkers would find the treasure with probability one, the expected (hitting) time becomes infinite.

Evaluating the running time as a function of D, the distance to the treasure, was studied in the context of the cow-path problem. Specifically, it was established in [5] that the competitive ratio for deterministically finding a point on the real line is nine, and that in the two-dimensional grid, the spiral search algorithm is optimal up to lower order terms. Several other varients where studied in [15, 36, 37, 38]. In particular, in [38], the cow-path problem was extended by considering k agents. However, in contrast to our setting, the agents they consider have unique identities, and the goal is achieved by (centrally) specifying a different path for each of the k agents.

The question of how important it is for individual processors to know their total number has recently been addressed in the context of locality. Generally speaking, it has been observed that for several classical local computation tasks, knowing the number of processors is not essential [26]. On the other hand, in the context of local decision, some evidence exist that such knowledge is crucial for non-deterministic decision [20].

2 Preliminaries

General setting: We consider the *Ants Nearby Treasure Search (ANTS)* problem initially introduced in [32]. In this *central place* searching problem, k mobile *agents* are searching for a *treasure* on the two-dimensional plane. The agents are probabilistic mobile machines (robots). They are identical, that is, all agents execute the same protocol $\mathcal{P}$. Each agent has some limited field of view, i.e., each agent can see its surrounding up to a distance of some $\varepsilon > 0$. Hence, for simplicity, instead of considering the two-dimensional plane, we assume that the agents are actually walking on the integer two-dimensional infinite grid $G = \mathbb{Z}^2$ (they can traverse an edge of the grid in both directions). The search is central place, that is, all k agents initiate the search from some central node $s \in G$, called the *source*. Before the search is initiated, an adversary locates the treasure at some node $t \in G$, referred to as the *target* node. Once the search is initiated, the agents cannot communicate among themselves. We denote by D the (Manhattan) distance between the source node and the target, i.e., $D = d_G(s, t)$. It is important to note that the agents have no a priori information about the location of t or about D. We say that the agents *find* the treasure when one of the agents visits the target node t. The goal of the agents it to find the treasure as fast as possible as a function of both D and k.

Since we are mainly interested in lower bounds, we assume a very liberal setting. In particular, we do not restrict neither the computational power nor the navigation capabilities of agents. Moreover, we put no restrictions on the internal storage used for navigation[1].

Oracles and Advice: We would like to model the situation in which before the search actually starts, some

[1]On the other hand, we note that for constructing upper bounds, the algorithms we consider use simple procedures that can be implemented using relatively little resources. For example, with respect to navigation, the constructions only assume the ability to perform four basic procedures, specifically: (1) choose a direction uniformly at random, (2) walk in a "straight line" to a prescribed distance and direction, (3) perform a *spiral search* around a given node (see, e.g., [5]), and (4) return to the source node.

initial communication may be made between the agents at the source node. In reality, this preliminary communication may be quite limited. This may be because of difficulties in the communication that are inherent to the agents or the environment, e.g., due to faults or limited memory, or because of asynchrony issues regarding the different starting times of the search, or simply because agents are identical and it may be difficult for agents to distinguish one agent from the other. Nevertheless, we consider a very liberal setting in which this preliminary communication is almost unrestricted.

More specifically, we consider a centralized algorithm called *oracle* that assigns advices to agents in a preliminary stage. The oracle, denoted by $\mathcal{O}$, is a probabilistic[2] centralized algorithm that receives as input a set of k agents and assigns an *advice* to each of the k agents. We assume that the oracle may use a different protocol for each k; given k, the randomized algorithm used for assigning the advices to the k agents is denoted by $\mathcal{O}_k$. Furthermore, the oracle may assign a different advice to each agent[3]. Observe, this definition of an oracle allows it to simulate almost any reasonable preliminary communication between the agents[4].

It is important to stress that even though all agents execute the same searching protocol, they may start the search with different advices. Hence, since their searching protocol may rely on the content of this initial advice, agents with different advices may behave differently. Another important remark concerns the fact that some part of the advices may be used for encoding (not necessarily disjoint) identifiers. That is, assumptions regarding the settings in which not all agents are identical and there are several types of agents can be captured by our setting of advice.

To summarize, a *search algorithm* is a pair $\langle \mathcal{P}, \mathcal{O} \rangle$ consisting of a randomized searching protocol $\mathcal{P}$ and randomized oracle $\mathcal{O} = \{\mathcal{O}_k\}_{k \in \mathbf{N}}$. Given k agents, the randomized oracle $\mathcal{O}_k$ assigns a separate advice to each of the given agents. Subsequently, all agents initiate the actual search by letting each of the agents execute protocol $\mathcal{P}$ and using the corresponding advice as input to $\mathcal{P}$. Once the search is initiated, the agents cannot communicate among themselves.

Consider an oracle $\mathcal{O}$. Given k, let $\Psi_{\mathcal{O}}(k)$ denote the maximum number of bits devoted for encoding the advice of an agent, taken over all coin tosses of $\mathcal{O}_k$, and over the k agents. In other words, $\Psi_{\mathcal{O}}(k)$ is the minimum number of bits necessary for encoding the advice, assuming the number of agents is k. Note that $\Psi_{\mathcal{O}}(k)$ also bounds from below the number of memory bits of an agent required by the search algorithm $\langle \mathcal{P}, \mathcal{O} \rangle$, assuming that the number of agents is k. The function $\Psi_{\mathcal{O}}(\cdot)$ is called the *advice size* function of oracle $\mathcal{O}$. (When the context is clear, we may omit the subscript $\mathcal{O}$ from $\Psi_{\mathcal{O}}(\cdot)$ and simply use $\Psi(\cdot)$ instead.)

Time complexity: When measuring the time to find the treasure, we assume that all internal computations are performed in zero time. For the simplicity of presentation, we assume that the movements of agents are synchronized, that is, each edge traversal is performed in precisely one unit of time. Indeed, this assumption can easily be removed if we measure the time according to the slowest edge-traversal. We also assume that all agents start the search simultaneously at the same time. This assumption can also be easily removed by starting to count the time when the last agent initiates the search.

[2]It is not clear whether or not a probabilistic oracle is strictly more powerful than a deterministic one. Indeed, the oracle assigning the advice is unaware of D, and may thus potentially use the randomization to reduce the size of the advices by balancing between the efficiency of the search for small values of D and larger values.

[3]We note that even though we consider a very liberal setting, and allow a very powerful oracle, the oracles we use for our upper bounds constructions are very simple and rely on much weaker assumptions. Indeed, these oracles are deterministic and assign the same advice to each of the k agents.

[4]For example, it can simulate to following very liberal setting. Assume that in the preprocessing stage, the k agents are organized in a clique topology, and that each agent can send a separate message to each other agent. Furthermore, even though the agents are identical, in this preprocessing stage, let us assume that agents can distinguish the messages received from different agents, and that each of the k agents may use a different probabilistic protocol for this preliminary communication. In addition, no restriction is made neither on the memory and computation capabilities of agents nor on the preprocessing time, that is, the preprocessing stage takes finite, yet unlimited, time.

The *expected running time* of a search algorithm $\mathcal{A} := \langle \mathcal{P}, \mathcal{O} \rangle$ is the expected time until at least one of the agents finds the treasure. The expectation is defined with respect to the coin tosses made by the (probabilistic) oracle $\mathcal{O}$ assigning the advices to the agents, as well as the subsequent coin tosses made by the agents executing $\mathcal{P}$. We denote the expected running time of an algorithm $\mathcal{A}$ by $\tau = \tau_{\mathcal{A}}(D, k)$. In fact, for our lower bound to hold, it is sufficient to assume that the probability that the treasure is found by time 2τ is at least $1/2$. By Markov inequality, this assumption is indeed weaker than the assumption that the expected running time is τ.

Note that if an agent knows D, then it can potentially find the treasure in time $O(D)$, by walking to a distance D in some direction, and then performing a circle around the source of radius D (assuming, of course, that its navigation abilities enable it to perform such a circle). On the other hand, with the absence of knowledge about D, an agent can find the treasure in time $O(D^2)$ by performing a spiral search around the source (see, e.g., [5]). The following observation imply that $\Omega(D + D^2/k)$ is a lower bound on the expected running time of any search algorithm. The proof is straightforward and can be found in [32].

Observation 2.1 *The expected running time of any algorithm is $\Omega(D+D^2/k)$, even if the number of agents k is known to all agents.*

We evaluate the time performance of an algorithm with respect to the lower bound given by Observation 2.1. Formally, let $\Phi(k)$ be a function of k. A search algorithm $\mathcal{A} := \langle \mathcal{P}, \mathcal{O} \rangle$ is called $\Phi(k)$-*competitive* if

$$\tau_{\mathcal{A}}(D, k) \leq \Phi(k) \cdot (D + D^2/k),$$

for every integers k and D. Our goal is establish connections between the *size* of the advice, namely $\Psi(k)$, and the competitiveness $\Phi(k)$ of the search algorithm.

More definitions: The *distance* between two nodes $u, v \in G$, denoted $d(u, v)$, is simply the Manhattan distance between them, i.e., the number of edges on the shortest path connecting u and v in the grid G. For a node u, let $d(u) := d(u, s)$ denote the distance between u and the source node. Hence, $D = d(t)$.

3 Lower Bounds on the Advice

The theorem below generalizes Theorem 4.1 in [32], taking into account the notion of advice. The proof of the theorem contains the main technical contribution of our paper. All our lower bound results follow as corollaries of this theorem. Note that for the theorem to be meaningful we are interested in advice size whose order of magnitude is less than $\log \log k$. Indeed, if $\Psi(k) = \log \log k$, then one can encode a 2-approximation of k in each advice, and obtain an optimal result, that is, an $O(1)$-competitive algorithm (see [32]).

Before stating the theorem, we need the following definition. A non-decreasing function $\Phi(x)$ is called *relatively-slow* if $\Phi(x)$ is sublinear (i.e., $\Phi(x) = o(x)$) and if there exist two positive constants c_1 and $c_2 < 2$ such that when restricted to $x > c_1$, we have $\Phi(2x) < c_2 \cdot \Phi(x)$. Note that this definition captures many natural sublinear functions[5].

Theorem 3.1 *Consider a $\Phi(k)$-competitive search algorithm using advice of size $\Psi(k)$. Assume that $\Phi(\cdot)$ is relatively-slow and that $\Psi(\cdot)$ is non-decreasing. Then there exists some constant x', such that for every $k > 2^{x'}$, the sum $\sum_{i=x'}^{\log k} \frac{1}{\Phi(2^i) \cdot 2^{\Psi(k)}}$ is at most some fixed constant.*

[5]For example, note that the functions of the form $\alpha_0 + \alpha_1 \log^{\beta_1} x + \alpha_2 \log^{\beta_2} \log x + \alpha_3 2^{\log^{\beta_3} \log x} \log x + \alpha_4 \log^{\beta_4} x \log^{\beta_5} \log x$, (for non-negative constants α_i and β_i, $i = 1, 2, 3, 4, 5$ such that $\sum_{i=1}^{4} \alpha_i > 0$) are all relatively-slow.

Proof. Consider a search algorithm with advice size $\Psi(k)$ and competitiveness $\Phi'(k)$, where $\Phi'(\cdot)$ is relatively-slow. By definition, the expected running time is less than $\tau(D,k) = (D+D^2/k)\cdot\Phi'(k)$. Note, for $k \leq D$, we have $\tau(D,k) \leq \frac{D^2\Phi(k)}{k}$, where $\Phi(k) = 2\Phi'(k)$, and $\Phi(\cdot)$ is relatively-slow. Let c_1 be the constant promised by the fact that Φ is relatively-slow. Let $x_0 > c_1$ be sufficiently large so that x_0 is a power of 2, and for every $x > x_0$, we have $\Phi(x) < x$ (recall, Φ is sublinear).

Fix an integer $T > x_0^2$. In the remaining of the proof, we assume that the treasure is placed somewhere at distance $D := 2T + 1$. Note, this means, in particular, that by time $2T$ the treasure has not been found yet. Next, for every integer $i \in [\log x_0, \frac{1}{2}\log T]$, set

$$k_i = 2^i \quad \text{and} \quad d_i = \sqrt{\frac{T \cdot k_i}{\Phi(k_i)}}.$$

Fix an integer i in $[\log x_0, \frac{1}{2}\log T]$, and let $B(d_i) := \{v \in G : d(v) \leq d_i\}$ denote the ball of radius d_i around the source node. We consider now the case where the algorithm is executed with k_i agents (using the corresponding advices given by the oracle $\mathcal{O}_{k_i}$). For every set of nodes $S \subseteq B(d_i)$, let $\chi_i(S)$ denote the random variable indicating the number of nodes in S that were visited by at least one of the k_i agents by time $2T$. (For short, for a singleton node u, we write $\chi_i(u)$ instead of $\chi_i(\{u\})$.) Note, the value of $\chi_i(S)$ depends on the values of the coins tosses made by the oracle for assigning the advices as well as on the values of the coins tossed by the k_i agents. Now, define the ring $R_i := B(d_i) \setminus B(d_{i-1})$. The proof of the following claim is deferred to Appendix A.

Claim 3.2 *For each integer $i \in [\log x_0, \frac{1}{2}\log T]$, we have $\mathbf{E}(\chi_i(R_i)) = \Omega(d_i^2)$.*

Note that for each $i \in [\log x_0 + 1, \frac{1}{2}\log T]$, the advice given by the oracle to any of the k_i agents must use at most $\Psi(k_i) \leq \Psi(\sqrt{T})$ bits. In other words, for each of these k_i agents, each advice is some integer whose value is at most $2^{\Psi(\sqrt{T})}$.

Let $W(j,i)$ denote the random variable indicating the number of nodes in R_i visited by the j'th agent by time $2T$, assuming that the total number of agents is k_i. By Claim 3.2, for every integer $i \in [\log x_0 + 1, \frac{1}{2}\log T]$, we have:

$$\mathbf{E}\left(\sum_{j=1}^{k_i} W(j,i)\right) \;\geq\; \mathbf{E}(\chi_i(R_i)) \;=\; \Omega(d_i^2).$$

By linearity of expectation, it follows that for every integer $i \in [\log x_0 + 1, \frac{1}{2}\log T]$, there exists an integer $j \in \{1, 2, \cdots, k_i\}$ for which

$$\mathbf{E}(W(j,i)) = \Omega(d_i^2/k_i) = \Omega(T/\Phi(k_i)).$$

Now, for each advice in the relevant range, that is, for each $a \in \{1, \cdots, 2^{\Psi(\sqrt{T})}\}$, let $M(a,i)$ denote the random variable indicating the number of nodes in R_i that an agent with advice a visits by time $2T$. Note, the value of $M(a,i)$ depends only on the values of the coin tosses made by the agent. On the other hand, note that the value of $W(j,i)$ depends on the results of the coin tosses made by the oracle assigning the advice, and the results of the coin tosses made by the agent that uses the assigned advice. Recall, the oracle may assign an advice to agent j according to a distribution that is different than the distributions used for other agents. However, regardless of the distribution used by the oracle for agent j, it must be the case that there exists an advice $a_i \in \{1, \cdots, 2^{\Psi(\sqrt{T})}\}$, for which $\mathbf{E}(M(a_i,i)) \geq \mathbf{E}(W(j,i))$. Hence, we obtain:

$$\mathbf{E}(M(a_i,i)) = \Omega(T/\Phi(k_i)).$$

Let $A = \{a_i \mid i \in [\log x_0 + 1, \frac{1}{2}\log T]\}$. Consider now an "imaginary" scenario[6] in which we execute the search algorithm with $|A|$ agents, each having a different advice in A. That is, for each advice $a \in A$, we have a different agent executing the algorithm using advice a. For every set S of nodes, let $\hat{\chi}(S)$ denote the random variable indicating the number of nodes in S that were visited by at least one of these $|A|$ agents by time $2T$, and let $\hat{\chi}$ denote the random variable indicating the total number of nodes that were visited by at least one of these agents by time $2T$.

By definition, for each $i \in [\log x_0 + 1, \frac{1}{2}\log T]$, the expected number of nodes in R_i visited by at least one of these $|A|$ agents is

$$\mathbf{E}(\hat{\chi}(R_i)) \geq \mathbf{E}(M(a_i, i)) = \Omega(T/\Phi(k_i)).$$

Since the sets R_i are pairwise disjoint, the linearity of expectation implies that the expected number of nodes covered by these agents by time $2T$ is

$$\mathbf{E}(\hat{\chi}) \;\geq\; \sum_{i=x_0+1}^{\frac{1}{2}\log T} \mathbf{E}(\hat{\chi}(R_i)) \;=\; \Omega\left(\sum_{i=x_0+1}^{\frac{1}{2}\log T} \frac{T}{\Phi(k_i)}\right) \;=\; T \cdot \Omega\left(\sum_{i=x_0+1}^{\frac{1}{2}\log T} \frac{1}{\Phi(2^i)}\right).$$

Recall that A is included in $\{1, \cdots, 2^{\Psi(\sqrt{T})}\}$. Hence, once more by linearity of expectation, there must exist an advice $\hat{a} \in A$, such that the expected number of nodes that an agent with advice $\hat{a}$ visits by time $2T$ is

$$T \cdot \Omega\left(\sum_{i=x_0+1}^{\frac{1}{2}\log T} \frac{1}{\Phi(2^i) \cdot 2^{\Psi(\sqrt{T})}}\right).$$

Since each agent may visit at most one node in one unit of time, it follows that, for every T large enough, the sum $\sum_{i=x_0+1}^{\frac{1}{2}\log T} 1/\Phi(2^i) \cdot 2^{\Psi(\sqrt{T})}$ is at most some fixed constant. The proof of the theorem now follows by replacing the variable T with T^2. □

Corollary 3.3 *Consider a $\Phi(k)$-competitive search algorithm using advice of size $\Psi(k)$. Assume that $\Phi(\cdot)$ is relatively-slow. Then, $\Phi(k) = \Omega(\log k/2^{\Psi(k)})$, or in other words, $\Psi(k) = \log\log k - \log\Phi(k) - O(1)$.*

Proof. Theorem 3.1 says that for every k, we have $\frac{1}{2^{\Psi(k)}} \sum_{i=1}^{\log k} \frac{1}{\Phi(2^i)} = O(1)$. On the other hand, since Φ is non-decreasing, we have $\sum_{i=1}^{\log k} \frac{1}{\Phi(2^i)} \geq \frac{\log k}{\Phi(k)}$. Hence, $\frac{\log k}{2^{\Psi(k)} \cdot \Phi(k)} = O(1)$. The corollary follows. □

The following corollary follows directly from the previous one.

Corollary 3.4 *Let $\epsilon < 1$ be a positive constant. Consider a $\frac{\log k}{2^{\log^\epsilon \log k}}$-competitive search algorithm using advice of size $\Psi(k)$. Then $\Psi(k) = \log^\epsilon \log k - O(1)$.*

Our next corollary implies that even though $O(\log\log k)$ bits of advice are sufficient for obtaining $O(1)$-competitiveness, roughly this amount of advice is necessary even for achieving relatively large competitiveness.

Corollary 3.5 *There is no search algorithm that is $O(\log^{1-\epsilon} k)$-competitive for some positive constant ϵ, using advice of size $\Psi(k) = \epsilon \log\log k - \omega(1)$.*

[6]The scenario is called imaginary, because, instead of letting the oracle assign the advice for the agents, we impose a particular advice to each agent, and let the agents perform the search with our advices. Note, even though such a scenario cannot occur by the definition of the model, each individual agent with advice a cannot distinguish this case from the case that the number of agents was some k' and the oracle assigned it the advice a.

Proof. Assume that the competitiveness is $\Phi(k) = O(\log^{1-\epsilon} k)$. Then, $\sum_{i=1}^{\log k} \frac{1}{\Phi(2^i)\cdot 2^{\Psi(k)}} = \Omega\left(\frac{\log^{\epsilon} k}{2^{\Psi(k)}}\right)$. According to Theorem 3.1, this sum must converge, and hence, we cannot have $\Psi(k) = \epsilon \log\log k - \omega(1)$. □

Corollary 3.6 *There is no $O(\log k)$-competitive search algorithm, using advice of size $\log\log\log k - \omega(1)$.*

Proof. Assume that the competitiveness is $\Phi(k) = O(\log k)$. Since $\Phi(2^i) = O(i)$, we have $\sum_{i=1}^{\log k} 1/\Phi(2^i) = \sum_{i=1}^{\log k} 1/i = \Omega(\log\log k)$. According to Theorem 3.1, $\frac{1}{2^{\Psi(k)}} \sum_{i=1}^{\log k} 1/\Phi(2^i) = \Omega(\log\log k/2^{\Psi(k)})$ must converge as k goes to infinity. In particular, we cannot have $\Psi(k) = \log\log\log k - \omega(1)$. □

4 Upper Bound

The lower bound on the advice size given in Corollary 3.5 is tight, as $O(\log\log k)$ bits of advice are sufficient to obtain an $O(1)$-competitive search algorithm. To further illustrate the power of Theorem 3.1, we now show that the lower bound mentioned in Corollary 3.6 is also tight. Theorem 1.2 follows by combining the theorem below and Corollary 3.6.

Theorem 4.1 *There exists an $O(\log k)$-competitive algorithm using $\log\log\log k + O(1)$ bits of advice.*

Proof. For each integer i, let $B_i := \{u : d(u) \leq 2^i\}$. Without loss of generality, we may assume that k is sufficiently large, specifically, $k \geq 4$. Given k agents, the oracle simply encodes the advice $O_k = \lfloor \log\log k \rfloor$ at each agent. Note that since $k \geq 4$, we have $O_k \geq 1$. Observe also that the advice O_k can be encoded using $\log\log\log k + O(1)$ bits.

For an advice α, let $K(\alpha)$ be the set of integers k such that k is a power of 2 and $O_k = \alpha$. Let $g(\alpha)$ be the number of elements in $K(\alpha)$, i.e., $g(\alpha) = |K(\alpha)|$. We enumerate the elements in $K(\alpha)$ from small to large, namely,

$$K(\alpha) = \{k_1(\alpha), k_2(\alpha), \cdots, k_{g(\alpha)}(\alpha)\},$$

where $k_\rho(\alpha)$ is the ρ's smallest integer in $K(\alpha)$, for $\rho \in \{1, 2, \cdots, g(\alpha)\}$. Consider Algorithm $\mathcal{A}$ described below.

The following lemma whose proof is deferred to Appendix B establishes the theorem.

Lemma 4.2 *Algorithm $\mathcal{A}$ is $O(\log k)$ competitive.* □

5 Conclusion and Discussion

As stated above, a central place allows for a preliminary stage in which a group of searchers may assess some knowledge about its size. From a biological perspective, very little is known about such processes. Our theoretical lower bounds on advice size may enable us to relate group search performance to the extent of information sharing within the nest. Furthermore, our lower bounds on the memory size (or, alternatively, on the number of states), may provide some evidence concerning the actual memory capacity of foraging ants.

A common problem, when studying a biological system is the complexity of the system and the huge number of parameters involved. A typical study involves a choice of the processes relevant for the behavior at hand, and the merging of these processes into a 'behavioral algorithm'. If the algorithm is simple enough, a small number of unknown parameters (e.g., chemical rates) may be tuned to simulate experimentally measured behaviors. It is not surprising that a large number of parameters could be used to fit the finite number

begin

Let α be the advice given to the agent. The agent first chooses an integer $\rho \in \{1, 2, \cdots, g(\alpha)\}$ uniformly at random, and performs the following double loop;

for j *from* 1 *to* ∞ **do** the *stage* j defined as follows

for i *from* 1 *to* j **do** the *phase* i defined as follows

- Let $t_i = 2^{2i+2^{\alpha}+2}/k_{\rho}(\alpha)$.
- Perform a spiral search (that starts at the source) for time $t_i + 2^i$.
- Return to the source s
- Go to a node $u \in B_i$ chosen uniformly at random among the nodes in B_i
- Perform a spiral search for time t_i.
- Return to the source s.

end

end

end

Algorithm 1: The algorithm $\mathcal{A}$.

of experimentally measured features. The methodology, therefore, stands to the test for its ability to predict system behavior in a different experimental setting. However, a different setting typically requires the consideration of more processes with new unknown parameters - and predictive strengths are restricted. To summarize, biological systems seem to resist these straightforward techniques.

Several alternative directions with the potential to provide concise are being explored. One tactic is reducing the parameter space. This is done by dividing the parameter space into critical and non-critical directions where changes in non-critical parameters do not affect overall system behavior [60, 61]. A different approach involves the definitions of bounds which govern a biological systems. Bounds may originate from physics: the sensitivity of eyes is limited by quantum shot noise and that of biochemical signaling pathways by noise originating from small number fluctuations [62]. Information theory has also been used to formulate such bounds, for example, by bounding information transmission rates of sensory neurons [63]. Note that the biological systems are confined by these bounds independently of any algorithms or parameters.

Our results are an attempt to draw non-trivial bounds on biological systems from the field of distributed computing. Such bounds are particularly interesting since they provide not a single bound but a relations between key parameters, in our case these would be the memory capacity of an agent and collective search efficiency. Indeed, suppose experiments on living ants indicate that they solve the search problem relatively fast [23]; viewing the memory of ants from an information theoretical point of view, such experiments can be combined with our memory lower bounds to give a quantitative evidence on the number of bits of memory used by ants for such tasks (or, alternatively, on the number of states they use). Obviously, to truly illustrate the concept, one must give precise (non-asymptotical) bounds and of course, conduct a careful experiment. This is beyond the scope of this paper. Nevertheless, we hope that our results may give some "proof of concept" that such a methodology is possible.

For conducting such an experiment, two natural candidates are desert ants *Cataglyphys* and honeybees *Apis mellifera*. One reason for focusing on these two species is that they both seem to face similar settings to the one we use. Indeed, communication appears to be negligible during the search because of the dispersedness of individuals [23] and their inability to leave chemical trails. Furthermore, the task of finding the treasure is relevant, as food sources in many cases are indeed relatively rare or patchy. Moreover, due to the reasons mentioned in Section 1, finding nearby sources of food is crucial. A second reason is that insects of these species seem to have computational power and behavioral patterns somewhat resembling the ones we use for

the algorithms described in the upper bounds (both in the current paper and in [32]). Indeed, such insects have been shown to have the computational capacity to maintain a compass-directed vector flight [13, 23], measure distance using an internal odometer [50, 51], travel to distances taken from a random power law distribution [47], and perform spiral or quasi-spiral movement patterns [44, 46, 55]. Moreover, the search trajectories of desert ants have been shown to include two distinguishable sections: a long straight path in a given direction emanating from the nest and a second more tortuous path within a small confined area [23, 54].

References

[1] Y. Afek, N. Alon, O. Barad, E. Hornstein, N. Barkai, and Z. Bar-Joseph. A biological solution to a fundamental distributed computing problem. Science, 331(6014):183–185, 2011.

[2] S. Albers and M. R. Henzinger. Exploring unknown environments. *SIAM J. on Computing*, (2000), 29, 1164–1188.

[3] N. Alon, C. Avin, M. Kouck, G. Kozma, Z. Lotker, and M. R. Tuttle. Many Random Walks Are Faster Than One. *Combinatorics, Probability & Computing* (2011), 20(4): 481–502.

[4] I. Averbakh and O. Berman. $(p-1)/(p+1)$-approximate algorithms for p-traveling salesmen problems on a tree with minmax objective. *Discr. Appl. Mathematics*, (1997), 75, 201–216.

[5] R. A. Baeza-Yates, J. C. Culberson and G.J.E. Rawlins. Searching in The Plane. *Information and Computation*, (1991), (106), (2), 234–252.

[6] M.A. Bender, A. Fernandez, D. Ron, A. Sahai and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proc. 30th Ann. Symp. on Theory of Computing (STOC)*, (1998), 269–278.

[7] V. Bonifaci, K. Mehlhorn, G. Varma. Physarum can compute shortest paths. SODA 2012: 233–240.

[8] H. Bockenhauer, D. Komm, R. Kralovic, and R. Kralovic. On the Advice Complexity of the k-Server Problem. ICALP (1) 2011: 207–218.

[9] N.J. Burroughs, M.de.O.B. Miguel Paz B and P.A. Adrego. Regulatory Tcell adjustment of quorum growth thresholds and the control of local immune responses. Journal of Theoretical Biology 241:134–141, 2006.

[10] J. Le Breton and V. Fourcassié. Information transfer during recruitment in the ant Lasius niger L. (Hymenoptera: Formicidae). Behavioral Ecology and Sociobiology 55(3):242–250, 2004.

[11] R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman and D. Peleg. Label-Guided Graph Exploration by a Finite Automation. ACM Transactions on Algorithms (TALG) 4(4), 2008.

[12] C. Cooper, A. M. Frieze, and T. Radzik. Multiple Random Walks in Random Regular Graphs. *SIAM J. Discrete Math*, (2009), 23(4), 1738-1761.

[13] E. A. Capaldi, A. D. Smith, J. L. Osborne, S. E. Fahrbach, S. M. Farris, D. R. Reynolds, A. S. Edwards, A. Martin, G. E. Robinson, G. M. Poppy and J. R. Riley. Ontogeny of orientation Flight in the honeybee revealed by harmonic radar. Nature 403. 537-540 (2000).

[14] B. Chazelle. Natural algorithms. In *Proc. Symp. Discrete Algorithms, (SODA)*, pages 422–431, 2009.

[15] E. D. Demaine, S. P. Fekete, and S. Gal. Online searching with turn cost. *Theor. Comput. Sci.* 361, 2 (September 2006), 342–355.

[16] R. Elsasser and T. Sauerwald. Tight bounds for the cover time of multiple random walks. *Theor. Comput. Sci. 412*,(2011), 24, 2623–2641.

[17] Y. Emek, P. Fraigniaud, A. Korman and A. Rosen. Online Computation with Advice. Theoretical Computer Science (TCS) , 412(24), (2011), 2642–2656.

[18] P. Fraigniaud, D. Ilcinkas, and A. Pelc. Oracle size: a new measure of difficulty for communication tasks. In PODC 2006, pages 179–187.

[19] P. Fraigniaud, C. Gavoille, D. Ilcinkas, and A. Pelc. Distributed computing with advice: information sensitivity of graph coloring. In ICALP 2007.

[20] P. Fraigniaud, A. Korman, and D. Peleg. Local Distributed Decision. FOCS 2011.

[21] P. Fraigniaud, A. Korman and E. Lebhar. Local MST Computation with Short Advice. Theory of Computing Systems (ToCS) , 47(4), (2010), 920–933.

[22] L.A. Giraldeau and T. Carco. Social Foraging Theory, 2000.

[23] R.D. Harkness and N.G. Maroudas. Central place foraging by an ant (*Cataglyphis bicolor* Fab.): a model of searching. Animal Behavior 33(3) 916–928, 1985.

[24] C.A. Janeway, P. Travers, M. Walport and M.J. Shlomchik Immunobiology: The Immune System in Health and Disease New Yoy: Garland Science, 2001.

[25] J. Krebs. Optimal foraging, predation risk and territory defense. *Ardea*, (1980), 68, 83–90. Nederlandse Ornithlogische Unie.

[26] A. Korman, J.S. Sereni, and L. Viennot. Toward More Localized Local Algorithms: Removing Assumptions Concerning Global Knowledge. *Proc. 30th ACM Symp. on Principles of Distributed Computing (PODC)*, 2011.

[27] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. How many oblivious robots can explore a line. *Inf. Process. Lett* (2011), 111(20): 1027–1031.

[28] A. Dessmark and A. Pelc. Optimal graph exploration without good maps. In *Proc. 10th European Symposium on Algorithms (ESA)*, (2002), LNCS 2461, 374–386.

[29] K. Diks, P. Fraigniaud, E. Kranakis and A. Pelc. Tree exploration with little memory. In *Proc. 13th Ann. ACM SIAM Symposium on Discrete Algorithms (SODA)*, (2002), 588–597.

[30] L. Gasieniec, A. Pelc, T. Radzik, X. Zhang. Tree exploration with logarithmic memory. In *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[31] O. Feinerman, G. Jentsch, K.E. Tkach, J.W. Coward, M.M. Hathorn, M.W. Sneddon, T. Emonet, K.A. Smith and G. Altan-Bonnet. Single-cell quantification of IL-2 response by effector and regulatory T cells reveals critical plasticity in immune response. Molecular systems biology. 6(437), 2010.

[32] O. Feinerman, A. Korman, Z. Lotker and J.S. Sereni. Collaborative Search on the Plane without Communication. To appear in *Proc. 31st ACM Symp. on Principles of Distributed Computing (PODC)*, 2012.

[33] P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc. Collective tree exploration. In *Proc. Latin American Theoretical Informatics (LATIN)*, (2004), LNCS 2976, 141–151.

[34] A. Korman and S. Kutten. Distributed Verification of Minimum Spanning Trees. Distributed Computing (DC) 20(4), 2007.

[35] A. Korman, S. Kutten and D. Peleg. Proof Labeling Schemes. Distributed Computing (DC) 22(4), 2010.

[36] M. Kao, J. H. Reif, and S. R. Tate. Searching in an Unknown Environment: An Optimal Randomized Algorithm for the Cow-Path Problem. *Journal of Inf. Comput.*, (1996), 63–79.

[37] R. M. Karp, M. Saks, and A. Wigderson. On a search problem related to branch-and-bound procedures. *In Proceedings of the 27th Annual Symposium on Foundations of Computer Science (FOCS)*, (1986), 19–28.

[38] A. López-Ortiz and G. Sweet. Parallel searching on a lattice. CCCG 2001: 125–128.

[39] G. F. Orians, and N. E. Pearson. On the theory of central place foraging. Analysis of Ecological Systems (pp. 155–177), 1979.

[40] P. Panaite and A. Pelc. Exploring unknown undirected graphs. *Journal of Algorithms*, (1999), 33, 281–295.

[41] M. M. Polycarpouy, Y. Yang, K. M. Passinoz. A Cooperative Search Framework for Distributed Agents. In *Intelligent Control.*, (2001), 1–6.

[42] S. Pratt, E. Mallon, D. Sumpter and N.R. Franks. Quorum sensing, recruitment, and collective decision-making during colony emigration by the ant Leptothorax albipennis. Behavioral Ecology and Sociobiology 52(2):117–127, 2002.

[43] A.M. Reynolds. Cooperative random Lévy flight searches and the flight patterns of honeybees. Physics Letters A 354 (2006) 384–388.

[44] A.M. Reynolds. Optimal random Lévy-loop searching: New insights into the searching behaviours of central-place foragers. European Physics Letters 82(2) 20001 (2008).

[45] O. Reingold. Undirected connectivity in log-space. *J. ACM* 55(4): (2008).

[46] A.M. Reynolds, A.D. Smith, M. Randolf, U. Greggers, D.R. Reynolds, and J.R. Riley. Displaced honey bees perform optimal scale-free search flights. Ecology 88(8) 1955–1961 (2007).

[47] A. M. Reynolds, A. D. Smith, D. R. Reynolds, N. L. Carreck and J. L. Osborne. Honeybees perform optimal scale-free searching flights when attempting to locate a food source. J. Exp Biol. 2007 Nov;210(Pt 21): 3763–3770.

[48] H.A. Rollik. Automaten in planaren graphen. Acta Informatica 13: 287–298, 1980.

[49] M.G. Surette, M.B. Miller and B.L. Bassler BL. Quorum sensing in Escherichia coli, Salmonella typhimurium, and Vibrio harveyi: a new family of genes responsible for autoinducer production. Proceedings National Acadamy of Science 96:1639–1644, 1999.

[50] S. Sommer and R. Wehner. The ant's estimation of distance travelled : experiments with desert ants, *Cataglyphis fortis*. Journal of Comparative Physiology A 190(1) 1–6. 2004.

[51] M. V. Srinivasan, S. Zhang, M. Altwein and J. Tautz. Honeybee Navigation: Nature and Calibration of the Odometer. Science 287: 851-853 (2000).

[52] P.C. Thomson. The behavioural ecology of dingoes in north-western Australia. III. Hunting and feeding behaviour, and diet. Wildlife Research 19: 531–542, 1992.

[53] C.D. Town, J.D. Gross and R.R. Kay. Cell differentiation without morphogenesis in Dictyostelium discoideum. Nature 262: 717–719, 1976.

[54] R. Wehner, C. Meier and C. Zollikofer. The ontogeny of foraging behaviour in desert ants, *Cataglyphis bicolor*. Ecol. Entomol. 29, 240–250 (2004).

[55] R. Wehner and M.Y. Srinivasan. Searching behaviour of desert ants, genus *Cataglyphis (Formicidae, Hymenoptera)* Journal of Comparative Physiology 142(3) 315–338 (1981)

[56] A. Zahavi. The function of pre-roost gatherings and communal roosts. Ibis 113: 106–109, 1971.

[57] C.J. Feare, G.M. Dunnet and I.J. Patterson. Ecologicalstudies of the rook (*Corvus frugilegus L.*) in north-east Scotland; Food intake and feeding behaviour. Journal of Applied Ecology 11: 867–896, 1974.

[58] D.M. Gordon. The regulation of foraging activity in red harvester ant colonies. The American Naturalist 159(5):509–518 ,2002.

[59] G. Berkolaiko and S. Havlin Territory covered by N Levy flights on d-dimensional lattices. Physical review. E 55(2): 1395–1400 , 1999.

[60] O. Feinerman, J. Veiga, J.R. Dorfman, R.N. Germain and G. Altan-Bonnet. Variability and robustness in T Cell activation from regulated heterogeneity in protein levels. Science 321(5892): 1081–1084, 2008.

[61] R.N. Gutenkunst, J.J. Waterfall, F.P. Casey, K.S. Brown, C.R. Myers and J.P. Sethna. Universally sloppy parameter sensitivities in systems biology models. PLOS Computational Biology 3(10): e189. doi:10.1371/journal.pcbi.0030189, 2007.

[62] W. Bialek. Physical limits to sensation and perception. Annual Review of Biophysics and Biophysical Chemistry, 16:455–478, 1987.

[63] F. Rieke, D. Warland and W. Bialek. Coding efficiency and information rates in sensory neurons. Europhysics Letters 22(2):15–156, 1993.

APPENDIX

A Proof of Claim 3.2

To see why the claim holds, note that by the properties of Φ, and from the fact that $2^i \leq \sqrt{T}$, we get that $k_i \leq d_i$, and therefore, $\tau(d_i, k_i) \leq \frac{d_i^2 \Phi(k_i)}{k_i} = T$. It follows that for each node $u \in B(d_i)$, we have $\tau(d(u), k_i) \leq T$, and hence, the probability that u is visited by time $2T$ is at least $1/2$, that is, $\mathbf{Pr}(\chi_i(u) = 1) \geq 1/2$. Hence, $\mathbf{E}(\chi_i(u)) \geq 1/2$. Now, by linearity of expectation,

$$\mathbf{E}(\chi_i(R_i)) = \sum_{u \in R_i} \mathbf{E}(\chi_i(u)) \geq |R_i|/2.$$

Consequently, by time $2T$, the expected number of nodes in R_i that are visited by the k_i agents is $\Omega(|R_i|) = \Omega\left(d_{i-1}(d_i - d_{i-1})\right) = \Omega\left(\frac{T \cdot k_i}{\Phi(k_{i-1})} \cdot \left(\sqrt{\frac{2\Phi(k_{i-1})}{\Phi(k_i)}} - 1\right)\right) = \Omega\left(\frac{T \cdot k_i}{\Phi(k_i)}\right) = \Omega(d_i^2)$, where the second equality follows from the fact that $d_i = d_{i-1} \cdot \sqrt{\frac{2\Phi(k_{i-1})}{\Phi(k_i)}}$, and the third equality follows from the fact that $\Phi(\cdot)$ is relatively-slow. This establishes the claim. □

B Proof of Lemma 4.2

Let us analyze the performances of algorithm $\mathcal{A}$. Our goal is to show that $\mathcal{A}$ is $O(\log k)$-competitive. We begin with the following observation.

Observation B.1 $g(O_k) = O(\log k)$.

To see why the observation holds, let $k_{\max}$ denote the maximum value such that $O_{k_{\max}} = O_k$, and let $k_{\min}$ denote the minimum value such that $O_{k_{\min}} = O_k$. We know, $g(O_k) \leq \log k_{\max}$. Since $O_{k_{\max}} = O_{k_{\min}}$, we have $\log \log k_{\max} < \log \log k_{\min} + 1$, and hence $\log k_{\max} < 2 \log k_{\min}$. The observation follows, as $k_{\min} \leq k$.

Observe now that there exists $\rho^* \in \{1, 2, \cdots, g(\alpha)\}$, such that $k_{\rho^*}(\alpha) \leq k < 2k_{\rho^*}(\alpha)$. Let N denote the random variable indicating the number of agents that choose ρ^*. Since each agent chooses an integer $\rho \in \{1, 2, \cdots, g(\alpha)\}$ uniformly at random, then the expected value of N is $\mathbf{E}(N) = k/g(\alpha) = \Omega(k/\log k)$. Let us now condition on the event that $N \geq \mathbf{E}(N)/2$. For the purposes of the proof, we consider for now only those N agents. Note, since these N agents choose ρ^*, then they execute Algorithm $\mathcal{A}$ with

$$t_i = \Theta(2^{2i} \cdot \frac{\log k}{k}) = \Omega(2^{2i}/N).$$

Recall that D denotes the distance from the treasure to the source. Let $s = \lceil \log D \rceil$. Fix a positive integer ℓ and consider the time T_ℓ until all the N agents completed ℓ phases i with $i \geq s$. Each time an agent performs phase i, the agent finds the treasure if the chosen node u belongs to the ball $B(v, \sqrt{t_i}/2)$ around the node v holding the treasure. Note that at least some constant fraction of the ball $B(v, \sqrt{t_i}/2)$ is contained in B_i. The probability of choosing a node u in that fraction is thus

$$\Omega(|B(v, \sqrt{t_i}/2)|/|B_i|) = \Omega(\log k/k),$$

which is at least β/N for some positive constant β. Thus, the probability that by time T_ℓ none of the N agents finds the treasure (while executing their respective ℓ phases i) is at most $(1 - \beta/N)^{N\ell}$, which is at most $\gamma^{-\ell}$ for some constant γ greater than 1.

For an integer i, let $\psi(i)$ be the time required (for one of the N agents) to execute a phase i. Note that $\psi(i) = O(2^i + 2^{2i}/N)$. Hence, the time elapsed from the beginning of the algorithm until all the N agents complete stage j_0 for the first time is

$$\sum_{j=1}^{j_0}\sum_{i=1}^{j}\psi(i) = O\left(\sum_{j=1}^{j_0}\left(2^j + \sum_{i=1}^{j}2^{2i}/N\right)\right) = O(2^{j_0} + 2^{2j_0}/N).$$

It follows that for any integer ℓ, all the N agents complete their respective stages $s+\ell$ by time $\hat{T}(\ell) = O(2^{s+\ell} + 2^{2(s+\ell)}/N)$. Observe that by this time, all these N agents have completed at least $\ell^2/2$ phases i with $i \geq s$. Consequently, the probability that none of the N agents finds the treasure by time $\hat{T}(\ell)$ is at most $\gamma^{-\ell^2/2}$. Hence, conditioning on the event that $N > \mathbf{E}(N)/2$, the expected running time is at most

$$O\left(\sum_{\ell=1}^{\infty}\hat{T}(l)\cdot\gamma^{-\ell^2/2}\right) = O\left(\sum_{\ell=1}^{\infty}\frac{2^{s+\ell}}{\gamma^{\ell^2/2}} + \frac{2^{2(s+\ell)}}{N\gamma^{\ell^2/2}}\right) = O\left(2^s + 2^{2s}/N\right) =$$

$$= O(D + D^2/N) = O(D + D^2/k)\cdot\log k.$$

On the other hand, by Chernoff inequality, we have:

$$\Pr[N \leq \mathbf{E}(N)/2] < e^{-E(N)/8} < e^{-\sqrt{k}}.$$

Since each agent performs a spiral search of size $t_i + 2^i$ around the source in each stage i, it follows that the treasure is found by time $O(D^2)$, with probability 1. Hence, all together, the expected running time is

$$O\left(D^2\cdot e^{-\sqrt{k}} + (D + D^2/k)\cdot\log k\right) = O\left((D + D^2/k)\cdot\log k\right).$$

In other words, $\Phi(k) = O(\log k)$, as desired. □